View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by DBIS EPub

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 110 (2017) 207–214

Procedia Computer Science

www.elsevier.com/locate/procedia

The 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017)

Enabling Tracks in Location-Based Smart Mobile Augmented Reality Applications

Rüdiger Pryss[a,*], Marc Schickler[a], Johannes Schobel[a], Micha Weilbach[a], Philip Geiger[a], Manfred Reichert[a]

[a] *Ulm University, Institute of Databases and Information Systems, James-Franck-Ring, Ulm, 89081, Germany*

Abstract

To assist users through contemporary mobile technology is demanded in a multitude of scenarios. Interestingly, more and more users crave for mobile assistance in their leisure time. Consequently, the number of mobile applications that support leisure activities increases significantly. Mobile augmented reality applications constitute an example for user assistance that is welcome in these scenarios. In the AREA (Augmented Reality Engine Application) project, we developed a kernel that enables sophisticated location-based mobile augmented reality applications. On top of this kernel, various projects were realized. In many of these projects, a feature to enable tracks was demanded. Tracks, for example, may assist users in the context of mountaineering. The development of an AREA algorithm that enables track handling requires new concepts that are presented in this paper. To demonstrate the performance of the developed algorithm, also results of an experiment are presented. As a lesson learned, mobile augmented reality applications that want to make use of the new algorithm can be efficiently run on present mobile operating systems and be effectively realized by engineers using the AREA framework. Altogether, the new track feature is another valuable step for AREA towards a comprehensive location-based mobile augmented reality framework.

© 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the Conference Program Chairs.

Keywords: Mobile Augmented Reality; Location-based Algorithms; Mobile Application Engineering; Track Handling

1. Introduction

More and more scenarios shall be assisted by mobile technology [1,2,3]. For example, when walking around in New York with its numerous tourist attractions, a smart mobile device shall detect these attractions to provide related information to them. In such a scenario, location-based mobile augmented reality is one appropriate concept to properly assist users. In the AREA project [4], we developed a kernel that enables sophisticated location-based mobile augmented reality applications. To be more precise, AREA detects predefined *points of interest* (POIs) within the camera view of a smart mobile device, positions them correctly, and provides additional information on the detected POIs. This additional information, in turn, is interactively provided to the user. Furthermore, an architecture was

* Corresponding author. Tel.: +49-731-5024136 ; fax: +49-731-5024134.
E-mail address: ruediger.pryss@uni-ulm.de

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the Conference Program Chairs.
10.1016/j.procs.2017.06.086

realized, which enables the rapid development of location-based mobile augmented applications on top of the kernel[4]. In general, AREA is used for various scenarios in everyday life[1] and it has revealed practicality in these projects. Recently, a feature for AREA that enables so-called tracks was often demanded. A track, in turn, may be used as the cycle path a user wants to perform in a certain area. Therefore, we developed a track algorithm for the AREA kernel that is presented in this work.

Section 2 presents fundamental aspects of the AREA framework. In Section 3, the coordinate system and the track notion used by AREA are introduced. The algorithm developed for the track handling, in turn, is presented in Section 4. A conducted experiment that evaluates the performance of the new track algorithm is presented in Section 5, while Section 6 discusses related work and Section 7 concludes the paper.

2. AREA Project

The AREA framework (cf. Table 1) basically consists of a mobile augmented reality kernel that enables location-based mobile augmented reality applications. In addition, it provides a sophisticated architecture to create location-based mobile augmented reality applications on top of the kernel. Four technical issues were crucial when developing the kernel. First, POIs must be correctly displayed even if the device is hold obliquely. Second, the approach to display POIs correctly must be provided efficiently to the user. To be more precise, even if multiple POIs are detected, the kernel shall display them without any delay. Third, the POI concept shall be integrated with common mobile operating systems (i.e., iOS, Android, and Windows Phone). Fourth, the kernel allows for the handling of points of interest clusters. So far, the AREA kernel mainly consists of three developed algorithms: The POI algorithm, the clustering algorithm, and the track algorithm. The first two algorithms were already presented (cf. Table 1;[4]).

Table 1. AREA Project

	AREA	**References**
Android App	√	5,6,7
iOS App	√	5,6,7
Windows Phone App	√	5,6,7
POI Algorithm (all mobile OS)	*AREARenderingPipeline Algorithm*	5,6,7,4
Clustering Algorithm (all mobile OS)	*AREACluster Algorithm*	4
Track Algorithm (all mobile OS)	*AREATrack Algorithm*	**this work**
Sensor Management (all mobile OS)	$SensorFusion(Compass, Gyroscope, Accelerometer)$	5,6,7,4
Architecture (all mobile OS)	$Version\ 2$	5,6,7,4
Overall Sensor Management Android	*Own approach*	5,6,7,4
Overall Sensor Management iOS	$Built-in\ functions\ OS$	5,6,7,4
Overall Sensor Management Windows Phone	$Built-in\ functions\ OS$	5,6,7,4

3. AREA Coordinate System and Track Notion

The concept of AREA to relate a user to the objects (i.e., POIs, tracks) detected in the camera view is based on five aspects. First, a virtual 3D world is used to relate the user's position to the one of the objects. Second, the user is located at the origin of this world. Third, instead of the physical camera, a virtual 3D camera is used that operates with the created virtual 3D world. The virtual camera is therefore placed at the origin of this world. Fourth, the different sensor characteristics of the supported mobile operating systems are covered to enable the virtual 3D world. On iOS, sensor data of the gyroscope and the accelerometer are used, whereas on Android sensor data of the gyroscope, the accelerometer and the compass of the mobile device are used to position the virtual 3D camera correctly. Fifth, the physical camera of the mobile device is adjusted to the virtual 3D camera based on the assessment of sensor data.

[1] See http://www.liveguide.de for mobile applications that use AREA.

In order to realize the presented concept, a coordinate system, consisting of three different sub-systems, is required. The first sub-system uses GPS, ECEF (Earth-Centered, Earth-Fixed), and ENU (East, North, Up) coordinates.[2] The second sub-system, in turn, uses a virtual 3D space with the user located at the origin. The third sub-system uses a virtual 3D camera, with the camera being again located at the origin of the 3D world.

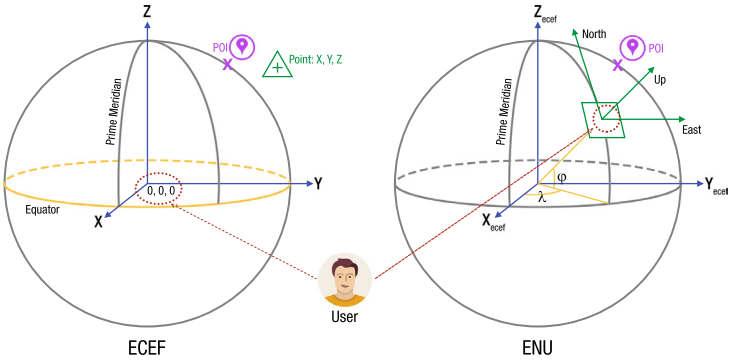

Fig. 1. ECEF and ENU Coordinate Systems

As illustrated in Fig. 1, the user is located at the ECEF origin $(0, 0, 0)$. The objects, in turn, are located on the surface of the earth, again using ECEF coordinates. To use this metaphor for the virtual 3D world, two additional transformations became necessary. As a smart mobile device can only sense GPS coordinates, first of all, the GPS coordinates of the user and objects need to be transformed into ECEF coordinates. Second, as a user cannot be physically located at the origin of the earth, ECEF coordinates are transformed into ENU coordinates. ENU coordinates, in turn, enable the required metaphor for the virtual 3D world. Finally, the distance between a user and the objects based on ENU coordinates can be calculated. Finally, Listing 1 presents the notion of tracks based on the presented coordinate system.

Listing 1. Track Notion (Android Version)

```
//Initialize list of vectors (x, y, z)
List<Float[]> pathStructure = new List<>();
//Set start point as origin
pathStructure.add([0.0, 0.0, 0.0]);

//Transform GPS-Coordinates into OpenGL-Coordinates
For each checkpoint{
   //Calculate angle at start point between this point and north axis (GPS)
   float alpha = calcNorthDegree(Location startPoint, Location currentCheckpoint)
   //Calculate vector
   float x = -OppositeLeg;
   float y = startPoint.altitude - currentCheckpoint.altitude;
   float z = AdjacentLeg;
}

//Generate rectangle between each consecutive point
List<Float[]> path = new List<>();

For each vector in pathStructure {
   //We cannot create a rectangle between only one point
   if !pathStructure.hasNext()
   break;
   //Calculate angle between this point and next point on xz-plane
   float angle = calcDegree(vector, vector.next());
   //Depending on angle calculate edges of rectangle
   List<Float[]> edges = calcEdges(vector, vector.next(), angle);
   //Triangulate rectangle for OpenGL renderer
   path.add(edges.get(0)); path.add(edges.get(1)); path.add(edges.get(2));
   path.add(edges.get(1)); path.add(edges.get(2)); path.add(edges.get(3));

   /*Generate curve at the end of the rectangle - if there would be a next rectangle,
```

[2] See https://en.wikipedia.org/wiki/ECEF and https://en.wikipedia.org/wiki/East_north_up

```
   a curve is generated as pitch circle consisting of several triangles.
   Next, determine degree between this rectangle and its next possible one */
   angle = calcNextDegree(vector.next(),vector.next().next());
   /* For each 5 degree interval, generate triangle and form the pitch circle.
   Thereby, a pitch circle starts on the last edge of the current rectangle and
   ends with the first edge of the next possible rectangle */
   List<Float[]> pitchCircle = calcCircle(edges.get(2), edges.get(3), angle);
   for each item in pitchCircle{
      path.add(item);
   }
}
```

4. Track Algorithm

Listing 2 presents the track calculation. Due to space limitations, solely the Android version of the algorithm is presented. The method *onDrawFrame* calculates the tracks and presents them to the user within the camera view. The method *onDrawFrame* is called multiple times per second, depending on the performance features of the smartphone. The core calculations are based on the following aspects: First, 3D models for all relevant tracks are calculated (cf. Listing 2, Line 9). As this calculation requires a lot of resources, all relevant tracks are precalculated. Relevance is given through several aspects. For example, only tracks are considered that run along a river. These aspects are not presented in more detail. After getting all relevant tracks, their 3D models are created (cf. Listing 2, Lines 10-12). Thereby, only those tracks are drawn that are inside a given radius (cf. Listing 2, Line 13). The latter can be specified by the user. The concrete positioning of tracks is performed as follows: Each track to be displayed is placed at the origin of the virtual world (cf. Listing 2, Lines 23-26). Following this, the current attitude of the smartphone is gathered and related to the origin in order to obtain the actual position of a track to be shown (cf. Listing 2, Lines 28-59). Finally, the track is drawn onto the camera view of the user (cf. Listing 2, Line 61).

Listing 2. Track Algorithm (Android Version)

```
1  public void onDrawFrame(GL10 gl) {
2     GLES20.glClearDepthf(1000.0f);              //Specifies object buffer depth for drawing area
3     GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT); //Empty drawing area
4
5     //Set rotated camera projection matrix to mMVPMatrix
6     //(e.g., http://www.songho.ca/opengl/gl_transform.html)
7     *Set mMVPMatrix
8
9     AREATrack[] t = getNewTracks();             //AREATrack[] stores all tracks
10        for(AREATrack aT : t){                  //Create 3D model for each detected track
11        //TOUR Class gets all track reference points and creates 3D track with fixed width
12        track TOUR nT = new TOUR(aT);
13            drawTOUR.add(nT);                   //drawTour stores all tracks within specified radius
14        }
15
16     TOUR[] dT = drawTOUR.toArray(dT);          //Renderer needs array instead of a list
17
18     for(TOUR aTour : dT) {                     //Track positioning on drawing area
19
20      //scra constitutes a 4x4 Matrix, which enables the correct positioning of a track
21      float[] scra = new float[16];
22
23      //setIdentity positions the track matrix mTranslationMatrix to the origin of the virtual 3D world
24      Matrix.setIdentityM(aTour.mTranslationMatrix, 0);
25      //translateM moves the track matrix mTranslationMatrix +1 around the y-axis
26      Matrix.translateM(aTour.mTranslationMatrix, 0, 0, 1, 0);
27
28      float[] rotateXMatrix = new float[16];    //4x4 matrix for rotation around the x-axis
29      float[] rotateYMatrix = new float[16];    //4x4 matrix for rotation around the y-axis
30      float[] rotateZMatrix = new float[16];    //4x4 matrix for rotation around the z-axis
31
32      //Positions track to center of the drawing area
33      Matrix.translateM(aTour.mTranslationMatrix, 0, -aTour.middleX, -aTour.middleY, -aTour.middleZ);
34
35      //Calculates rotation matrix (i.e., z-axis) based on smartphones rotation; calcRotations() determines the rotation
36      Matrix.setRotateM(rotateZMatrix, 0, calcRotation(), 0, 0, 1);
37      //Calculates rotation matrix (i.e., y-axis) based on viewing direction of the smartphone;
38      //newKnownDegreeX determines the viewing direction
39      Matrix.setRotateM(rotateYMatrix, 0, newKnownDegreeX, 0, 1, 0);
```

```
    //Calculates rotation matrix (i.e., x-axis) based on inclination of smartphone; perspective determines the inclination
    Matrix.setRotateM(rotateXMatrix, 0, perspective, 1, 0, 0);

    //Combines rotation matrix for y-axis rotateYMatrix with track matrix mTranslationMatrix
    Matrix.multiplyMM(aTour.mTranslationMatrix, 0, rotateYMatrix, 0, aTour.mTranslationMatrix, 0);
    //Combines rotation matrix for x-axis rotateXMatrix with track matrix mTranslationMatrix
    Matrix.multiplyMM(aTour.mTranslationMatrix, 0, rotateXMatrix, 0, aTour.mTranslationMatrix, 0);

    //Undo -> Positions track to center of the drawing area
    Matrix.translateM(aTour.mTranslationMatrix, 0, aTour.middleX, aTour.middleY, aTour.middleZ);

    //calcTourTranslation() determines the distance of a track to the position of the user based on GPS coordinates
    float[] xyzTranslation = calcTourTranslation(AREAVariables.currentLocation, aTour.startPointGPS);
    //Combines rotation matrix for z-axis rotateYMatrix with track matrix mTranslationMatrix
    Matrix.multiplyMM(aTour.mTranslationMatrix, 0, rotateZMatrix, 0, aTour.mTranslationMatrix, 0);

    //Undo -> ranslateM moves the track matrix mTranslationMatrix +1 around the y-axis
    Matrix.translateM(aTour.mTranslationMatrix, 0, 0, -1, 0);
    //Combines track matrix mTranslationMatrix with mMVPMatrix
    Matrix.multiplyMM(scra, 0, mMVPMatrix, 0, aTour.mTranslationMatrix, 0);

    aTour.draw(scra);                          //4x4 matrix for rotation around the x-axis

   }
}
```

Fig. 2 illustrates how track handling is presented to the user on Android and iOS. The first two screenshots are impressions of algorithm results on iOS, whereas the last two screenshots are impressions of the algorithm result on Android. Note that the area algorithm is not shown in this work.

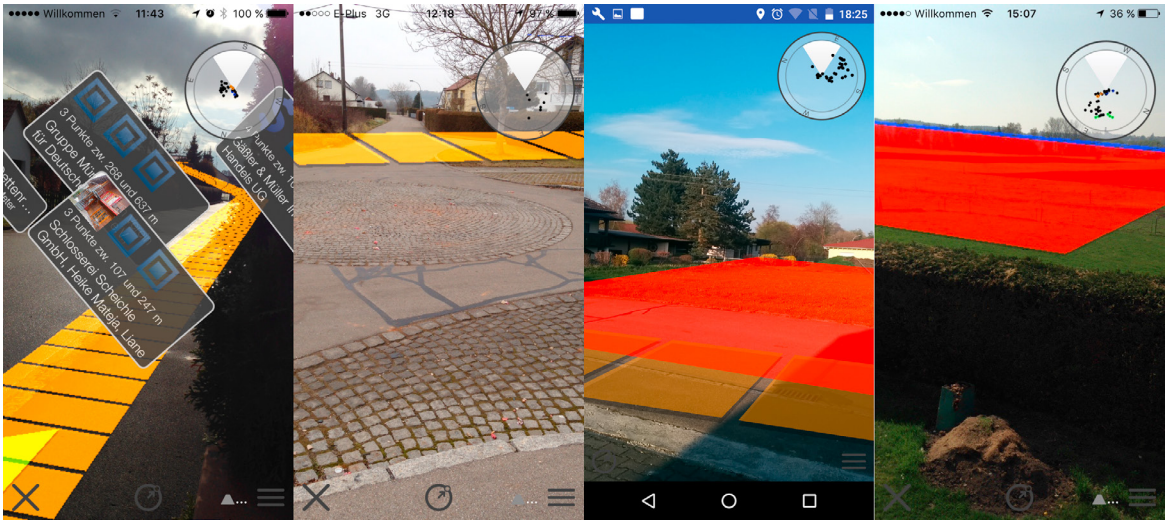

Fig. 2. Track Algorithm in Practice

5. Experimental Results

In order to evaluate various performance indicators of the AREA track algorithm, we compare these performance issues with the ones of competitive location-based mobile augmented reality applications. Therefore, we conducted an experiment, obeying the following goals. First, we specified the following performance indicators for the Android and the iOS version of AREA: CPU usage, memory usage, and battery consumption. Note that we omit the experiment for Windows Phone. Second, we compare these indicators with the ones of well-known smart mobile applications providing location-based mobile augmented reality including track handling as well. Based on this, we defined an experiment setting for using the smart mobile devices in two different scenarios: **(a)**, when holding the smart mobile device without performing any position change and **(b)**, when continuously moving the smart mobile device.

For the concrete experiment, we use an Apple iPhone 5c for the AREA iOS version and a Google Nexus 5 for the AREA Android version. Concerning the mobile applications AREA is compared to, we determined 3D World Path[8] on iOS and AR GPS Navigator[9] on Android as the competitive applications. In addition, we determine performance indicators for the camera as well as the main menu of the two smart mobile devices. Thereby, camera means that solely the camera function of the smart mobile device was started without using a particular smart mobile application. Main menu, in turn, means that the main menu of AREA was opened without using the augmented function. These two measurements were accomplished to enable a better comparison of the three performance indicators.

Concerning the two scenarios **(a)** and **(b)**, the following experimental setting was established: for the static Scenario (a), a vice was used (cf. Fig. 3; left pictures) to simulate a user holding the smart mobile device without any position change. For simulating a user continuously moving his smart mobile device (Scenario (b)), we used a ventilator (cf. cf. Fig. 3; right pictures). Note that for Scenario (b) the user himself is not moving, but based on the movement of the ventilator, the number of detected POIs is continuously changing. Consequently, a moving user can be simulated.

Fig. 3. Simulation of (a) Static Scenario (Pictures 1,2) and (b) Dynamic Scenario (Pictures 3,4)

For properly measuring the above mentioned three performance indicators, we used the SystemPanel App[10] for Android and the Instruments framework[11] for iOS. Based on this overall setting, each application was evaluated using the same experiment procedure:

- The smart mobile device was set to factory defaults and the battery was loaded to 100%.
- The smart mobile application and the monitoring app were downloaded.
- All other mobile user applications (i.e., except the background processes) were terminated.
- The smart mobile device was mounted to the vice or ventilator.
- The two mobile applications (i.e., test and monitoring application) were started.
- The experiment was conducted over a period of 30 minutes.

Table 2 shows the results of the experiment. For each tested application, the average value of a performance indicator during the 30-minutes experiment is shown. Note that the three applications AREA, 3D World Path and AR GPS Navigator provide alike location-based mobile augmented reality functions.

Experimental results indicate that AREA shows a better performance than the tested commercial location-based mobile augmented reality application AR GPS Navigator and competes well with 3D World Path. Regarding the battery performance indicator, AREA performs best in all scenarios, except for the static scenario on Android. This issue will be tackled in more detail in further developments and experimental results. Regarding the RAM indicator, AREA shows better results on Android than the commercial tool, whereas on iOS it shows slightly weaker results. Therefore, this is another issue that is currently dealt with. Regarding the CPU indicator, AREA is better on Android, while on iOS, it is comparable to the commercial application for the moving scenario and shows weaker results for the static scenario. This is the third issue that is tackled in current developments. Overall, 12 comparative results of AREA with the two commercial tools providing track features were shown. Thereby, seven times AREA showed

Table 2. Experimental Results

Device	Scenario	Application	CPU	RAM	Battery
iPhone 5c	Static (a)	AREA	71,61%	86,00%	17,00%
iPhone 5c	Static (a)	3D World Path	62,00%	72,00%	21,00%
iPhone 5c	Static (a)	Camera	30,36%	81,00%	13,00%
iPhone 5c	Static (a)	Main Menu	14,34%	53,00%	0,00%
iPhone 5c	Moving (b)	AREA	72,10%	86,00%	26,00%
iPhone 5c	Moving (b)	3D World Path	70,25%	72,00%	43,00%
iPhone 5c	Moving (b)	Camera	30,36%	81,00%	13,00%
iPhone 5c	Moving (b)	Main Menu	14,34%	53,00%	0,00%
Nexus 5	Static (a)	AREA	55,07%	28,00%	54,00%
Nexus 5	Static (a)	AR GPS Navigator	68,74%	53,00%	24,00%
Nexus 5	Static (a)	Camera	22,37%	49,00%	14,00%
Nexus 5	Static (a)	Main Menu	8,91%	41,00%	6,00%
Nexus 5	Moving (b)	AREA	57,21%	52,00%	28,00%
Nexus 5	Moving (b)	AR GPS Navigator	62,58%	54,00%	33,00%
Nexus 5	Moving (b)	Camera	22,37%	49,00%	14,00%
Nexus 5	Moving (b)	Main Menu	8,91%	41,00%	6,00%

better results, one time AREA showed slightly weaker results, and 4 times AREA showed weaker results. Altogether, the track algorithm developed for AREA revealed meaningful results.

6. Related Work

Previous research related to the development of a location-based augmented reality application in non-mobile environments is described in [12]. In turn, [13] uses smart mobile devices for developing an augmented reality system. The augmented reality application described in [14] allows sharing media data and other information in a real-world environment and enables users to interact with this data through augmented reality. However, none of these approaches share insights regarding the development of location-based augmented reality on smart mobile devices as AREA does. Regarding tracks in mobile augmented reality, only little work can be found. For example, the approaches [15,16,17] present tracks as key feature of (mobile) augmented reality applications. However, algorithms for implementing track handling are not presented. Moreover, only little work exists, which deals with the engineering of mobile augmented reality systems in general. As an exception, [18] validates existing augmented reality browsers. Moreover, [19] discusses various types of location-based augmented reality scenarios. More precisely, issues that have to be particularly considered for a specific scenario are discussed in more detail. However, engineering issues of mobile applications are not considered. In [20], an authoring tool for mobile augmented reality applications, which is based on marker detection, is proposed. In turn, [21] presents an approach for indoor location-based mobile augmented reality. Furthermore, [22] gives an overview of various aspects of mobile augmented reality for indoor scenarios.

Another scenario for mobile augmented reality is presented in [23]. The authors use mobile augmented reality for image retrieval. However, [20,21,22,23] do not address engineering aspects of location-based mobile applications. In [24], an approach supporting pedestrians with location-based mobile augmented reality is presented. Finally, [25] deals with a client and server framework enabling location-based applications. Altogether, neither software vendors nor research projects provide insights regarding the engineering of a location-based mobile augmented reality kernel, which includes track handling.

7. Summary and Outlook

This paper gave insights into the implementation details of the track algorithm for the AREA kernel. On top of the presented algorithm, the paper showed experimental results of the algorithm. To be more precise, an experiment with two commercial tools have been carried out and results be presented. The experimental results demonstrated that the track algorithm of the AREA kernel had shown a good performance compared to competitive location-based mobile augmented reality applications. In general, AREA shows that a mobile augmented reality kernel providing sophisticated features like track handling can be accomplished on different mobile operating systems. In future, new features will be addressed for the AREA kernel. For example, we currently integrate a trail navigation for mountainous regions. Altogether, mobile augmented reality shows that mobile applications become more and more mature. On the other, suitable concepts are required to deal with the complex peculiarities of contemporary mobile technology.

References

1. Schobel, J., Schickler, M., Pryss, R., Maier, F., Reichert, M.. Towards Process-Driven Mobile Data Collection Applications: Requirements, Challenges, Lessons Learned. In: *10th Int'l Conf on Web Information Systems and Technologies*. 2014, p. 371–382.
2. Schobel, J., et al. Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned. In: *9th Int'l Conf on Web Information Systems and Technologies (WEBIST 2013)*. 2013, p. 509–518.
3. Schobel, J., Pryss, R., Reichert, M.. Using smart mobile devices for collecting structured data in clinical trials: Results from a large-scale case study. In: *IEEE 28th Int'l Symposium on Computer-Based Medical Systems*. 2015, .
4. Pryss, R., Geiger, P., Schickler, M., Schobel, J., Reichert, M.. Advanced algorithms for location-based smart mobile augmented reality applications. *Procedia Computer Science* 2016;**94**:97–104.
5. Schickler, M., Pryss, R., Schobel, J., Reichert, M.. An engine enabling location-based mobile augmented reality applications. In: *10th Int'l Conf on Web Information Systems and Technologies (Revised Selected Papers)*; no. 226 in LNBIP. Springer; 2015, p. 363–378.
6. Geiger, P., Schickler, M., Pryss, R., Schobel, J., Reichert, M.. Location-based mobile augmented reality applications: Challenges, examples, lessons learned. In: *10th Int'l Conf on Web Information Systems and Technologies*. 2014, p. 383–394.
7. Geiger, P., Pryss, R., Schickler, M., Reichert, M.. Engineering an advanced location-based augmented reality engine for smart mobile devices. Technical Report UIB-2013-09; University of Ulm; 2013.
8. 3D World Path. `https://itunes.apple.com/ca/app/world-path-3d/id592730575?mt=8`; 2017. [Online; accessed 07-April-2017].
9. AR GPS Navigator. `https://play.google.com/store/apps/details?id=com.w.argps&hl=de`; 2017. [Online; accessed 07-April-2017].
10. Systempanel. `https://play.google.com/store/apps/details?id=nextapp.systempanel.r1&hl=de`; 2017. [Online; accessed 07-April-2017].
11. Instruments. `https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/`; 2017. [Online; accessed 07-April-2017].
12. Kooper, R., MacIntyre, B.. Browsing the real-world wide web: Maintaining awareness of virtual information in an ar information space. *Int'l Journal of Human-Computer Interaction* 2003;**16**(3):425–446.
13. Kähäri, M., Murphy, D.. Mara: Sensor based augmented reality system for mobile imaging device. In: *5th IEEE and ACM Int'l Symp on Mixed and Augmented Reality*; vol. 13. 2006, .
14. Lee, R., Kitayama, D., Kwon, Y., Sumiya, K.. Interoperable augmented web browsing for exploring virtual media in real space. In: *Proc of the 2nd Int'l Workshop on Location and the Web*. ACM; 2009, p. 7.
15. Vlahakis, V., Karigiannis, J., Tsotros, M., Ioannidis, N., Stricker, D.. Personalized augmented reality touring of archaeological sites with wearable and mobile computers. In: *Sixth International Symposium on Wearable Computers*. IEEE; 2002, p. 15–22.
16. Lee, T., Hollerer, T.. Hybrid feature tracking and user interaction for markerless augmented reality. In: *Virtual Reality Conference*. IEEE; 2008, p. 145–152.
17. Hollerer, T.. *User interfaces for mobile augmented reality systems*. Ph.D. thesis; Columbia University; 2004.
18. Grubert, J., Langlotz, T., Grasset, R.. Augmented reality browser survey. Technical Report; Graz University of Technology; 2011.
19. Kim, W., Kerle, N., Gerke, M.. Mobile augmented reality in support of building damage and safety assessment. *Natural Hazards and Earth System Sciences* 2016;**16**(1):287.
20. Yang, Y., et al. Mobile augmented reality authoring tool. In: *10th Int'l Conf on Semantic Computing*. IEEE; 2016, p. 358–361.
21. Paucher, R., Turk, M.. Location-based augmented reality on mobile phones. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE; 2010, p. 9–16.
22. Reitmayr, G., Schmalstieg, D.. Location based applications for mobile augmented reality. In: *Proc of the Fourth Australasian user interface conference on User interfaces*. Australian Computer Society, Inc.; 2003, p. 65–73.
23. Lee, Y., Rhee, S.. Efficient photo image retrieval system based on combination of smart sensing and visual descriptor. *Intelligent Automation & Soft Computing* 2015;**21**(1):39–50.
24. Chung, J., Pagnini, F., Langer, E.. Mindful navigation for pedestrians: Improving engagement with augmented reality. *Technology in Society* 2016;**45**:29–33.
25. Capece, N., Agatiello, R., Erra, U.. A client-server framework for the design of geo-location based augmented reality applications. In: *20th Int'l Conf on Information Visualisation*. IEEE; 2016, p. 130–135.